

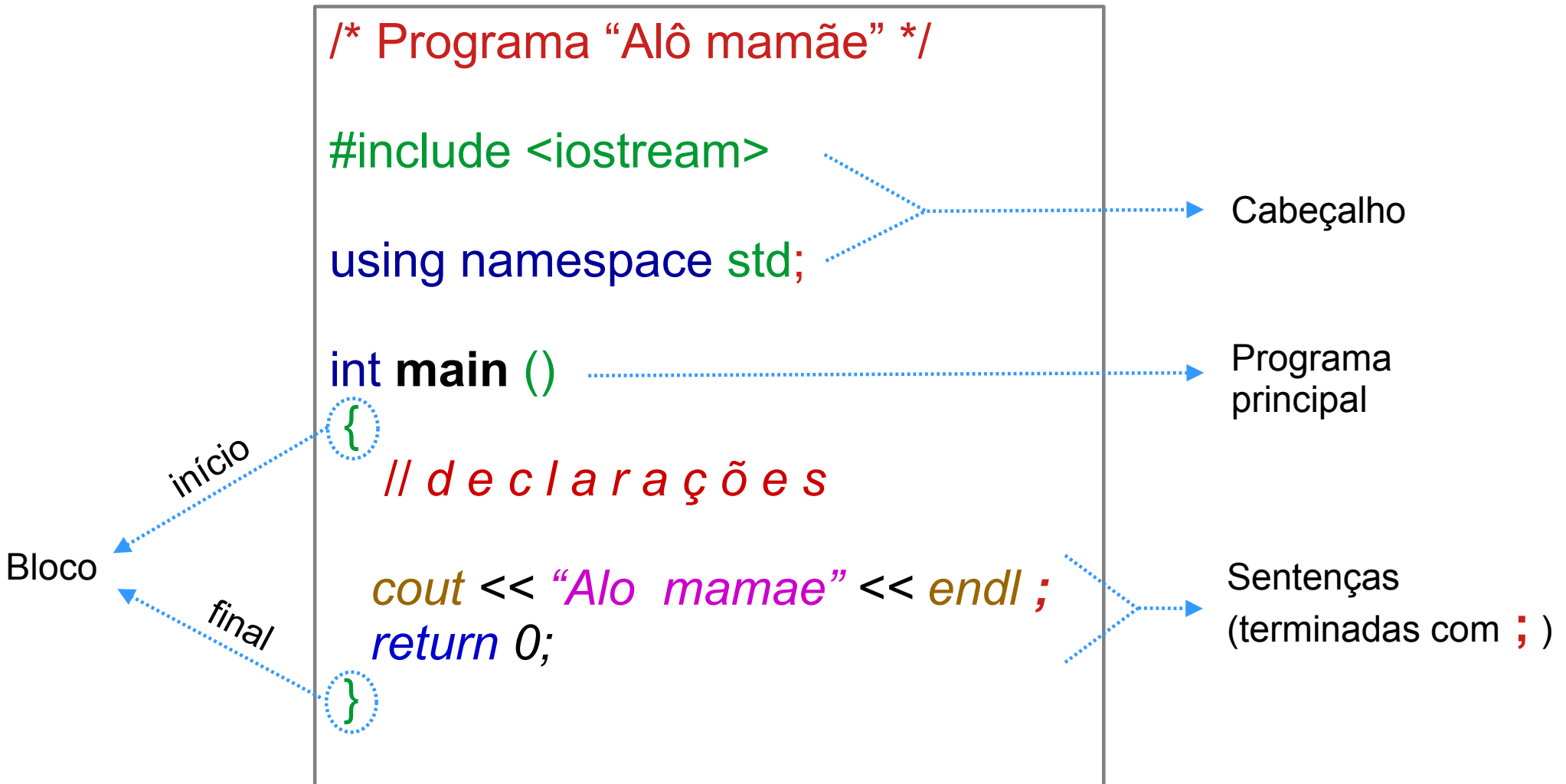
Primeiros Programas

Linguagem C++, Variáveis,
Atribuição de valores,
Expressões Aritméticas,
Entrada e saída

Sumário

- Estrutura de um programa
- Elementos da linguagem C++
- Variáveis e Tipos de dados
- Comando de atribuição
- Operações aritméticas
- Leitura e escrita

Estrutura de um Programa



Elementos da linguagem

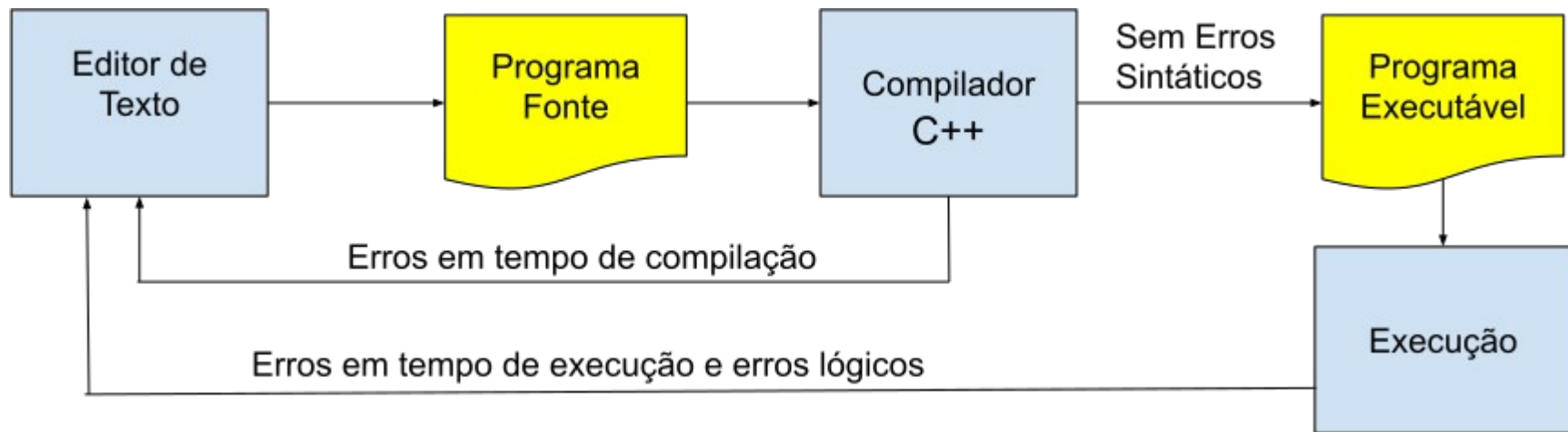
```
/* Programa "Alô mamãe"  
   Autora: Carmem  
*/  
  
#include <iostream>  
  
using namespace std;  
  
int main ()  
{  
    // declarações  
  
    cout << "Alo mamae" << endl;  
    return 0;  
}
```

- Palavras reservadas
- Funções pré-definidas
- Identificadores:
 - ▷ **letra** seguida de sequência de letras, números e ‘_’
 - ▷ palavras reservadas → NÃO
 - ▷ operações da linguagem → NÃO
 - ▷ letras acentuadas → NÃO
- Constantes: numéricas, sequência de caracteres
- Comentários
 - */* [texto em várias linhas] */*
 - *// [texto em uma única linha]*

Processo de compilação

Código-fonte → Programa executável

```
/* Programa "Alô mamãe" */  
  
#include <iostream>  
  
using namespace std;  
  
int main ()  
{  
    /* declarações */  
  
    cout << "Alo mamae" << endl;  
    return 0;  
}
```



Processo de compilação

The screenshot shows the Code::Blocks IDE with a C++ program named 'alomamae.cpp'. The code is as follows:

```
1  /* Programa "Alô mamãe"
2  */
3  #include <iostream>
4
5  using namespace std;
6
7  int main ()
8  {
9      /* declarações */
10
11     cout << "Alo mamae" << endl;
12 }
13
```

Below the code editor, the 'Logs & others' window shows the compilation output:

```
File      Line  Message
==== Build file: "no target" in "no project" (compiler: unknown) ====
==== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ====
```

Janela do Code::Blocks

Janela do resultado
(aparece ao executar
o programa)

The terminal window shows the execution output of the program:

```
Alo mamae
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Tipos de Erros

Linha onde compilador "percebeu" um erro

```
1  /* Programa "Alô mamãe"
2  */
3  #include <iostream>
4
5  using namespace std;
6
7  int main ()
8  {
9      /* declarações */
10
11     cout << "Alo mamae" << endl
12 }
13
```

Logs & others

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknow...
/home/nicolui/...		In function 'int main()':
/home/nicolui/...	12	error: expected ';' before '}' token
		=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 s...

/home/nicolui/doc/UFPR/Graduacao/ci208/20.1/Perio... C/C++ Unix (LF) UTF-8 Line 13, Col 1, Pos 156 Insert Read/Wri... default

Descrição de erro de compilação

Variáveis

- Elementos usados para guardar valores em um programa
 - ▷ Cada variável representa um espaço em memória
 - ▷ TIPO de valor a representar → quantas posições de memória
 - ▶ inteiro, real, caracter, etc.
 - ▷ DECLARADAS antes das sentenças do programa (normalmente...)

```
/* Programa "Peso da Idade" */
#include <iostream>
using namespace std;

int main ()
{
    /* declarações */
    int idade;
    char inicial;
    float peso;
    bool casado;
    .....
}
```

Memória RAM



- **letra** seguida de sequência de letras, números e `'_'`
- palavras reservadas → NÃO
- sinais da linguagem → NÃO
- letras acentuadas → NÃO

Variáveis

Declaração de Variáveis

tipo *identificador1* , *identificador2* , ... ;

Separa os nomes de variáveis

Termina (finaliza) declaração.

Tipo	Exemplo
int	10, -15, 0, 3, -6
float, double	1.5, 15.7, -23.8, 1.5e3
char	'a', 'B', '2'
bool	1 (true), 0 (false)

Variáveis

Faixas de valores possíveis para cada tipo de variável

Tipo	Tamanho na memória (bytes)	Faixa de valores
int	4	-2147483648 → 2147483647
float	4	1.5e-45 → 3.4e38
double	8	5.0e-324 → 1.7e308
char	1	-128 → 127
bool	1	0 (false) 1 (true)

Constantes numéricas, Literais e *Strings*

- Para expressar valores fixos como números e textos:
 - ▷ **Constantes** para valores numéricos: 3, 4.5, -23
 - ▷ **Literais** para caracteres de texto: 'a', 'B', '0', '1'
 - Delimitados por ***apóstrofes***
 - ▷ ***Strings*** pra representar textos: “Meu nome eh Armando”
 - Delimitados por **aspas**

Comando de Atribuição

Sintaxe

variável = expressão ;

- Usado para definir o valor de uma variável
 - **variável** RECEBE valor de **expressão**

```
#include <iostream>
using namespace std;

int main ()
{
    int idade;
    float peso;
    char inicial;
    bool casado;

    idade = 32;
    inicial = 'a';
    peso = 45.5;
    casado = true;
    .....
}
```

Operadores Aritméticos

Para inteiros

- [unário]
* / % (MOD)
+ -

- `id = 10 * 2;`
- `id = id + 1;`
- `negativo = -id;` → **-21**

-[unário] → troca sinal do operando

Para reais

- [unário]
* /
+ -

- `peso = -7.5;`
- `peso = peso + 2.5;`
- `peso = peso / 2;` → **-2.5**

Temos um operador diferente: %

O operador aritmético %

- Muitas vezes deseja-se fazer operações de divisão com números **inteiros**
 - ▷ Recuperar o **quociente** e o **resto** da divisão
 - ▶ $23 / 5 \rightarrow$ quociente **4**, resto **3** $\rightarrow 23 - (4 * 5) = 3 \rightarrow 23 = (4 * 5) + 3$
 - ▶ Como colocar em variáveis diferentes os valores **4** e **3**?
 - ▷ Resposta:
 - ▶ Operador de **divisão** (**/**) com operandos **inteiros** \rightarrow **quociente**
 - ▶ Operador de **módulo da divisão** (**%**) com operandos **inteiros** \rightarrow **resto**

```
/* Programa "Quociente+Resto" */
#include <iostream>
using namespace std;

int main ()
{
    int q, r;

    q = 23 / 5;           // q ← 4
    r = 23 - q * 5;      // r ← 3
    cout >> q >> " " >> r >> endl;
}
```



```
/* Programa "Quociente+Resto" */
#include <iostream>
using namespace std;

int main ()
{
    int q, r;

    q = 23 / 5;           // q ← 4
    r = 23 % 5;          // r ← 3
    cout >> q >> " " >> r >> endl;
}
```

Precedência de Operadores Aritméticos

```
/* Programa "Precedencias" */
#include <iostream>
using namespace std;

int main ()
{
    float a, b, c;

    a = 10 * 2;
    b = 7.5 / 2.8;
    c = 2.0;

    b = b * 2.0 + a;    // * antes de +
    b = b * (2.0 + a); // + antes de *
    b = a / c * 3;     // / antes de *
    b = a / (c * 3);   // * antes de /
    .....
}
```

Precedência de operadores:
prioridade de aplicação de operações

MAIOR
precedência

()
- [unário]
* / %
+ -

Lembrando que o operador %
(resto da divisão) só é válido
para operandos **inteiros**

Escrita

Comandos de escrita: exibe valores na tela (saída)

`cout << expr1 << expr2 << << exprn ;` [exibe e cursor fica na mesma linha]

`cout << expr1 << expr2 << << exprn << endl ;` [exibe e cursor muda de linha]

ATENÇÃO: Exibe os valores um após o outro, SEM ESPAÇO entre eles

```
/* Programa "Escrita" */
#include <iostream>
using namespace std;

int main ()
{
    cout << "Segunda " ;
    cout << "Aula " ;
    cout << "de CI208 " ;
    cout << "Quarta , dia " << 5 << endl;
    return 0;
}
```


Escrita

The image shows a screenshot of a C++ IDE (Code::Blocks 17.12) with a file named 'escrita.cpp'. The code in the editor is as follows:

```
1  /* Programa "Escrita"
2  */
3  #include <iostream>
4
5  using namespace std;
6
7  int main ()
8  {
9      cout << "Segunda ";
10     cout << "Aula ";
11     cout << " de CI208 ";
12     cout << "Quarta, dia " << 5 << endl;
13 }
14
```

Overlaid on the IDE is a terminal window titled 'Running' showing the program's output:

```
Segunda Aula de CI208 Quarta, dia 5
Process returned 0 (0x0)   execution time : 0.005 s
Press ENTER to continue.
```

At the bottom of the IDE, the 'Logs & others' panel shows a message:

```
F: L: Message
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

The status bar at the bottom indicates the current file path, encoding (UTF-8), and cursor position (Line 11, Col 17, Pos 145).

Leitura

Comandos de leitura: lê valores para variáveis (entrada)

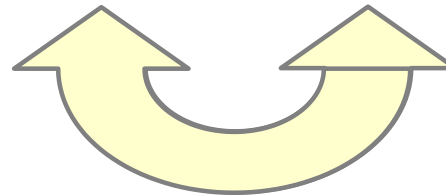
```
cin >> var1 >> var2 >> .... >> varn ;
```

```
/* Programa "Leitura" */  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int anoNasc;  
    cin >> anoNasc;  
    cout << "Em 31/12/2020 voce terah " , 2020 - anoNasc << " anos." << endl;  
    return 0;  
}
```

Leitura

```
/* Programa "Leitura" */  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int valor1, valor2;  
  
    cin >> valor1 >> valor2;  
  
    cout << "Media = " << (valor1+valor2) / 2.0  
        << endl;  
    return 0;  
}
```

```
/* Programa "Leitura" */  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int valor1, valor2;  
  
    cin >> valor1;  
    cin >> valor2;  
  
    cout << "Media = " << (valor1+valor2) / 2.0  
        << endl;  
    return 0;  
}
```



As duas formas de leitura são equivalentes

Exercícios para aula *online*

Após assistir todas as vídeo-aulas da semana, procure trabalhar na **Lista de exercícios** do Tópico **Primeiros Programas**, na sala virtual da disciplina na UFPR Virtual.

Estes exercícios serão usados nas aulas *online* para esclarecer e consolidar os conceitos abordados até aqui.

Leitura complementar

Acesse o **Material complementar** do Tópico **Primeiros Programas**, na sala virtual da disciplina da UFPR Virtual.

Elas são importantes e auxiliam na compreensão dos temas abordados até aqui.

Créditos: O conteúdo original deste documento é de autoria da Prof^a Carmem Satie Hara (DINF/ET), e foi adaptado pelo Prof. Armando L.N. Delgado (DINF/ET) para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183)

Compartilhe este documento de acordo com a licença abaixo



Este documento está licenciado com uma Licença *Creative Commons Atribuição-NãoComercial-SemDerivações* 4.0 Internacional.
<https://creativecommons.org/licenses/by-nc-sa/4.0/>